对券商和保险板块分别运用银行翻倍策略对2014年-2019年2月的数据进行排序然后设置相应的仓位权重，得出以下结果。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 策略年化 | 基准年化 | Alpha | 最大回撤 |
| 券商-无择时 | 19.58% | 7.66% | 0.13 | 56% |
| 券商-择时 | 29.52% | 7.66% | 0.24 | 45% |
| 保险-无择时 | 25.17% | 23.98% | 0.06 | 43% |
| 保险-择时 | 35.19% | 23.98% | 0.2 | 43% |
| 组合（市值加权） | 22.59% | 13.5% | 0.13 | 24% |

1.券商板块选取10只股，经过银行翻倍策略排序以后，设置仓位权重

最好收益的公式是结合ps和营业收入增长率。

1.1无择时版本：



1.2有择时版本



**2.保险板块选取所有成分股，经过银行翻倍策略排序以后，设置仓位权重**

最好收益的公式是结合ps和营业利润增长率。

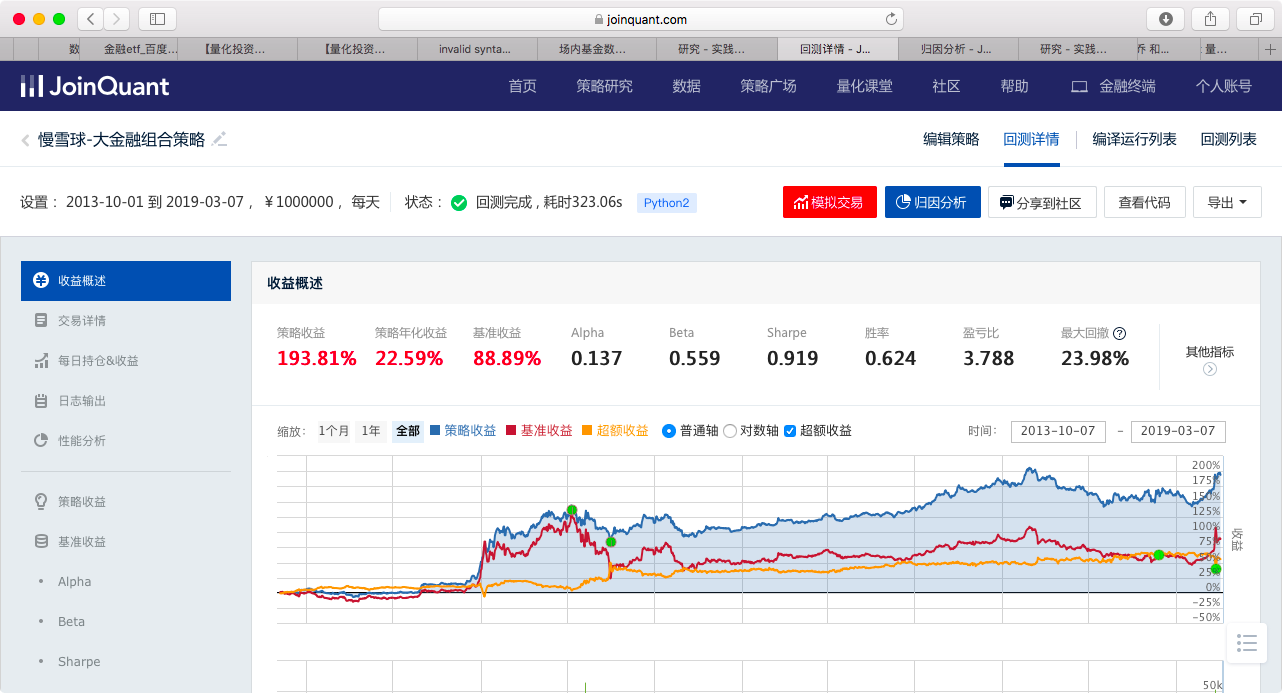
2.1无择时：



2.2 有择时：



3.大金融组合-银行5/保险4/券商6-市值加权/无择时

策略采用250天的roe均值，以及最新pb/ps值。

1. **import** statsmodels.api as sm
2. **from** statsmodels **import** regression
3. **import** numpy as np
4. **import** pandas as pd
5. **import** time
6. **from** datetime **import** date
7. **from** jqdata **import** \*
8. **import** datetime
9. **from** dateutil.relativedelta **import** relativedelta
10. **import** matplotlib.pyplot as plt
11. **from** statsmodels.tsa.stattools **import** adfuller
12. **from** six **import** StringIO

15. #总体回测前要做的事情
17. **def** initialize(context):
18. #context.zeshixinhao = ['2015-04-08']  # 择时空仓信号时间
20. #set\_option('futures\_margin\_rate', 1)
21. set\_params()
22. set\_backtest()
24. # 获得所有股票
25. run\_monthly(handle\_group, 1,'before\_open')
27. set\_order\_cost(OrderCost(open\_tax=0, close\_tax=0, open\_commission=0.0003, close\_commission=0.0013, close\_today\_commission=0, min\_commission=0), type='stock')
29. #1
31. #设置策参数
33. **def** set\_params():
35. g.banknum = 5
36. g.secnum = 6
37. g.insnum = 4
38. #2
40. #设置回测条件
42. **def** set\_backtest():
44. set\_option('use\_real\_price', True) #用真实价格交易
46. log.set\_level('order', 'error')
48. set\_benchmark('510230.XSHG')
50. set\_slippage(FixedSlippage(0.02))     #将滑点设置为0
52. #6
53. **def** handle\_group(context):
55. buylist\_bank,bank\_r = bank\_stock(context)
56. buylist\_sec,sec\_r = sec\_stock(context)
57. buylist\_ins,ins\_r = ins\_stock(context)
58. buylist = buylist\_bank+buylist\_sec+buylist\_ins
59. rlist = list(bank\_r)+list(sec\_r)+list(ins\_r)
61. #权重计算-市值加权
62. q = query(indicator.code,valuation.circulating\_market\_cap).filter(indicator.code.in\_(buylist))
63. df = get\_fundamentals(q, context.previous\_date.strftime('%Y-%m-%d')).set\_index('code')
64. weight = df['circulating\_market\_cap']/df['circulating\_market\_cap'].sum()
65. '''''
66. #权重计算-r值加权
67. weight = rlist/sum(rlist)
68. log.info(weight)
69. '''
70. **for**  stock **in** context.portfolio.positions:
71. **if** stock **not** **in** buylist:
72. order\_target\_value(stock, 0)
74. **for** stock,w **in** zip(buylist,weight):
75. order\_target\_value(stock, context.portfolio.total\_value\*w)

78. **def** bank\_stock(context):
80. stock\_list = get\_industry\_stocks('801192')
81. stock\_list = filter\_specials(stock\_list,context)
82. df=pd.DataFrame(index =stock\_list, columns =['ROE','PB','growth\_rate'])
83. **for** stk1 **in** stock\_list:
84. P = get\_fundamentals(query(indicator.code, valuation.pb\_ratio,indicator.inc\_operation\_profit\_year\_on\_year).filter(indicator.code==stk1))
85. RO = get\_fundamentals\_continuously(query(indicator.code, indicator.roe).filter(indicator.code == stk1),count=250)
86. df['PB'][stk1]=P['pb\_ratio'].values
87. df['ROE'][stk1] = RO['roe'].values.mean()/100
88. df['growth\_rate'][stk1] = P['inc\_operation\_profit\_year\_on\_year'].values/100
90. df = df[(df['ROE'] > -1)&(df['growth\_rate'] > -1)&(df['PB'] > 0)]


94. df["double\_time"] =  df.apply(**lambda** row: round(math.log(2.0 \* row['PB']/(1+row['growth\_rate']) , (1.0+row['ROE'])),2), axis=1)
95. #翻倍期最小的银行股
96. df = df.sort("double\_time")
98. #输出r值
99. r\_list = df["double\_time"][:g.banknum].values
100. #r\_list = ZscoreNormalization(r\_list)
102. **print** context.current\_dt.strftime('%Y-%m-%d') +' 银行选股为 '+ str(df.index[:5].values)[1:-1]
103. **return** df.index[:g.banknum],r\_list#这里的数字表示买入的标的 数量，默认是5
105. **def** sec\_stock(context):
107. stock\_list = get\_industry\_stocks('801193')
108. stock\_list = filter\_specials(stock\_list,context)
109. df=pd.DataFrame(index =stock\_list, columns =['ROE','PS','growth\_rate'])
110. **for** stk1 **in** stock\_list:
111. P = get\_fundamentals(query(indicator.code, valuation.ps\_ratio,indicator.inc\_revenue\_year\_on\_year).filter(valuation.code==stk1))
112. RO = get\_fundamentals\_continuously(query(indicator.code, indicator.roe).filter(valuation.code==stk1),count=250)
113. df['PS'][stk1]=P['ps\_ratio'].values
114. df['ROE'][stk1] = RO['roe'].values.mean()/100
115. df['growth\_rate'][stk1] = P['inc\_revenue\_year\_on\_year'].values/100
117. df = df[(df['ROE'] > -1)&(df['growth\_rate'] > -1)&(df['PS'] > 0)]

120. df["double\_time"] =  df.apply(**lambda** row: round(math.log(2.0 \* row['PS']/(1+row['growth\_rate']) , (1.0+row['ROE'])),2), axis=1)
121. #翻倍期最小的银行股
122. df = df.sort("double\_time")
124. #输出r值
125. r\_list = df["double\_time"][:g.secnum].values
126. #r\_list = ZscoreNormalization(r\_list)
128. **print** context.current\_dt.strftime('%Y-%m-%d') +' 券商选股为 '+ str(df.index[:5].values)[1:-1]
129. **return** df.index[:g.secnum],r\_list#这里的数字表示买入的标的 数量，默认是5
131. **def** ins\_stock(context):
133. stock\_list = get\_industry\_stocks('801194')
134. stock\_list = filter\_specials(stock\_list,context)
135. df=pd.DataFrame(index =stock\_list, columns =['ROE','PS','growth\_rate'])
136. **for** stk1 **in** stock\_list:
137. P = get\_fundamentals(query(indicator.code, valuation.ps\_ratio,indicator.inc\_operation\_profit\_year\_on\_year).filter(valuation.code==stk1))
138. RO = get\_fundamentals\_continuously(query(indicator.code, indicator.roe).filter(valuation.code==stk1),count=250)
139. df['PS'][stk1]=P['ps\_ratio'].values
140. df['ROE'][stk1] = RO['roe'].values.mean()/100
141. df['growth\_rate'][stk1] = P['inc\_operation\_profit\_year\_on\_year'].values/100
143. df = df[(df['ROE'] > -1)&(df['growth\_rate'] > -1)&(df['PS'] > 0)]

146. df["double\_time"] =  df.apply(**lambda** row: round(math.log(2.0 \* row['PS']/(1+row['growth\_rate']) , (1.0+row['ROE'])),2), axis=1)
147. #翻倍期最小的银行股
148. df = df.sort("double\_time")
150. #输出r值
151. r\_list = df["double\_time"][:g.insnum].values
152. #r\_list = ZscoreNormalization(r\_list)
154. **print** context.current\_dt.strftime('%Y-%m-%d') +' 保险选股为 '+ str(df.index[:5].values)[1:-1]
155. **return** df.index[:g.insnum],r\_list#这里的数字表示买入的标的 数量，默认是5


159. #过滤退市，停牌，ST
160. **def** filter\_specials(stock\_list,context):
161. curr\_data = get\_current\_data()
162. stock\_list = [stock **for** stock **in** stock\_list **if** \
163. (**not** curr\_data[stock].paused)  # 未停牌
164. **and** (**not** curr\_data[stock].is\_st)  # 非ST
165. **and** ('ST' **not** **in** curr\_data[stock].name)
166. **and** ('\*' **not** **in** curr\_data[stock].name)
167. **and** ('退' **not** **in** curr\_data[stock].name)
168. **and** (curr\_data[stock].low\_limit < curr\_data[stock].day\_open < curr\_data[stock].high\_limit)                   ]
169. **return** stock\_list
171. **def** ZscoreNormalization(x):
172. """Z-score normaliaztion"""
173. x = (x - np.mean(x)) / np.std(x)
174. **return** x
176. '''''
177. def GrahamStockSelect(context):
179. stocks = list(get\_all\_securities(['stock']).index) # 全部上市A股
181. #调用函数获取股票代码、eps、营业利润增长率存入df
183. q = query(indicator.code,indicator.eps,indicator.inc\_operation\_profit\_year\_on\_year,valuation.circulating\_market\_cap).filter(indicator.code.in\_(stocks))
185. df = get\_fundamentals(q, context.previous\_date.strftime('%Y-%m-%d')).set\_index('code')
186. #调用函数获取前一交易日收盘价
188. df1 = history(1, unit='1d', field='close', security\_list=stocks)
190. df1 = df1.T #因为得到的df1是以股票为列索引，所以将数组转置
192. df1.columns=['close'] #收盘价那一列重命名为close
194. df = df.join(df1) #两个数组合并


198. #格氏成长股公式
200. df['value'] = df['eps']\*(8.5+2\*df['inc\_operation\_profit\_year\_on\_year'])
202. df['outvalue\_ratio'] = df['value']/df['close']


206. #将结果从大到小排序，最后选前30只股票
208. df.sort("outvalue\_ratio",inplace = True, ascending = False)
210. df = df[:100]


214. df.sort("circulating\_market\_cap",inplace=True,ascending=True)
216. df = df[:30]
218. g.buylist = df.index
220. for  stock in context.portfolio.positions:
221. if stock not in g.buylist:
222. order\_target\_value(stock, 0)
223. for stock in g.buylist:
224. order\_target\_value(stock, g.ratio\_inbank\*context.portfolio.total\_value/len(g.buylist))
226. '''